MESH : Enabling Scalable Social Group Analytics Via Hyper Graph Analysis Systems

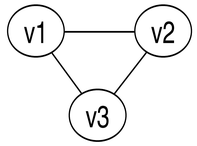
Students: Benjamin Heintz, Janani Thirunavukkarasu, Jayapriya Surendran, Shivangi Singh
PIs: Abhishek Chandra, Jaideep Srivastava (University Of Minnesota, Twin Cities)
URL: MESH.CS.UMN.EDU
Sponsor : National Science Foundation

MOTIVATION

Rapid growth of social data: Likes. Tweets, Publications

Need to transform data into knowledge

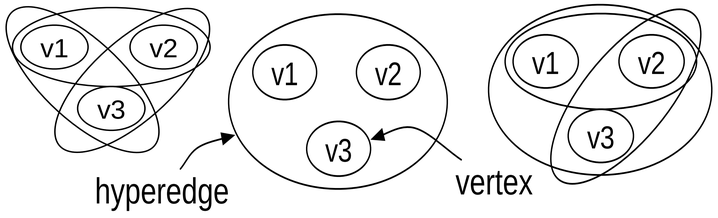
- Importance / centrality / influence
- Community detection, Shortest paths
- Information flow



State of the art: Graph Computational Systems

- Pregel, GraphLab (Dato), Apache Spark GraphX

To better model social **group** structure and behavior, we need **hypergraph** computing systems.



MESH : PROTOTYPE

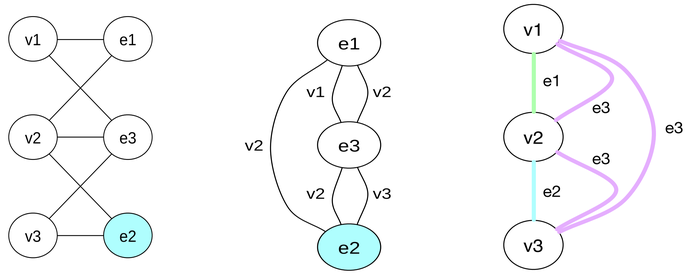
```
trait HyperGraph[HVD, HED] {
  def compute[ToE, ToV](
    maxIters: Int,
    initialMsg: ToV,
    hvProgram: Program[HVD, ToV, ToE],
    heProgram: Program[HED, ToE, ToV])
    : HyperGraph[HVD, HED]
}

object HyperGraph {
  trait Program[A, InMsg, OutMsg] {
    def messageCombiner: MessageCombiner[OutMsg]
    def procedure: Procedure[A, InMsg, OutMsg]
  }

  type MessageCombiner[Msg] = (Msg, Msg) => Msg

  type Procedure[A, InMsg, OutMsg] =
    (Int, NodeId, A, InMsg, Context[A, OutMsg]) => Unit

  trait Context[A, OutMsg] {
    def become(attr: A): Unit
    def send(msgF: NodeId => OutMsg,
             to: Recipients): Unit
  }
}
```

Iterative computation

Vertex ***and hyperedge*** programs

Message flow:
- vertex → hyperedge
- hyperedge → vertex

CHALLENGES: REPRESENTATION

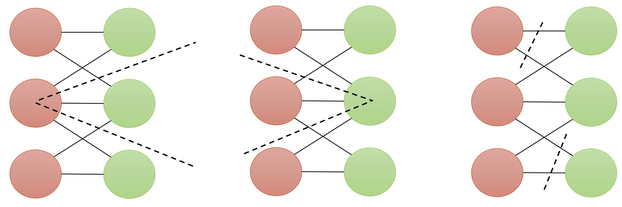


Bipartite graph
- ⊘ Obscures differences between hyperedges, vertices
- ✓ Portable to any graph system

Multigraph
- ⊘ Limited system support
- ⊘ Hard to implement with existing APIs
- ✓ Can exploit differences between vertices, hyperedges

CHALLENGES: PARTITIONING

Leveraging existing partitioning methods:



Vertex cut **Hyperedge cut** **Edge cut – 1D/2D**

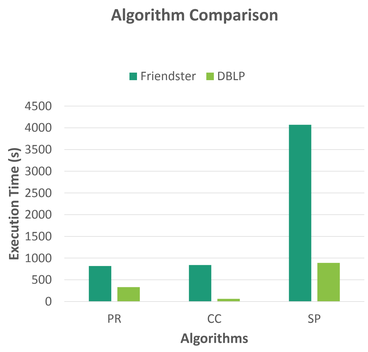
Partitioning for load balancing.

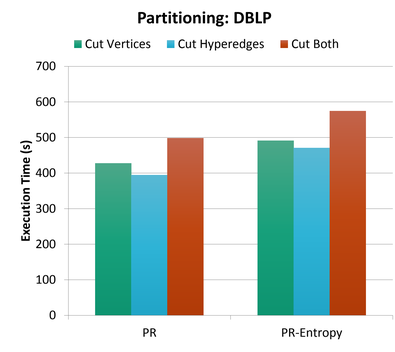
Sources of imbalance in hypergraph:
- Dataset – Ratio of vertices to hyperedges
- Skewed degree distribution –vertex or hyperedge or both
- Computation – vertex Vs hyperedge program
- Interaction – between vertices and hyperedges

RESULTS

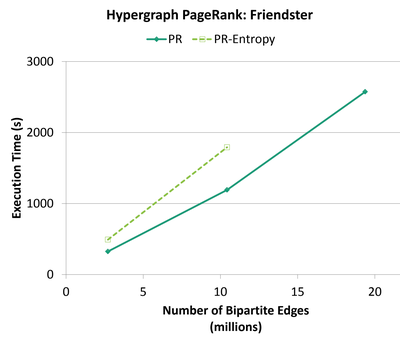
Proof-of-concept prototype
- Implemented on Apache Spark GraphX 1.2.1
- Run on shared 6-node cluster (2x6-core, 24GB RAM each)
- Using bipartite graph representation

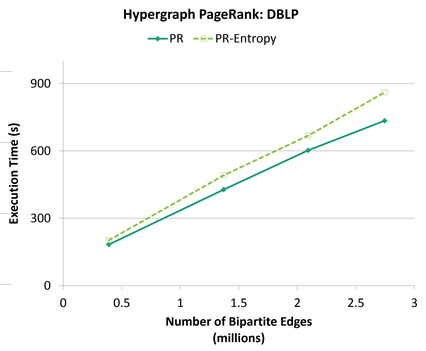
Dataset	Vertices	Hyperedges	Bipartite Edges	1-mode Projection Edges
DBLP	952,115 authors	916,947 collaborations	2,768,930	21,592,883
Friendster	7,944,949 users	1,620,991 communities	23,479,217	> 15.1 B





Performance heavily affected by
- Dataset Characteristics + Algorithm Characteristics
- Representation + Partitioning





Scalability is a real challenge.